Efficient Application Integration in IP-Based Sensor Networks

Dogan Yazar, Adam Dunkels
Swedish Institute of Computer Science
{dogan,adam}@sics.se

Abstract

Sensor networks are seen as an important part in emerging office and building energy management system, but the integration of sensor networks with future energy management systems is still an open problem. We present an IP-based sensor network system where nodes communicate their information using Web services, allowing direct integration in modern IT systems. Our system uses two mechanisms to provide a good performance and low-power operation: a session-aware power-saving radio protocol and the use of the HTTP Conditional GET mechanism. We perform an extensive evaluation of our system and show that Web services are a viable mechanism for use in low-power sensor networks. Our results show that Web service requests can be completed well below one second and with a low power consumption, even in a multi-hop setting.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Applications

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Sensor networks, web services, REST

1 Introduction

Sensor networks are seen as an important part in the emerging fields of energy management for homes, offices, and the smart grid. Many existing sensor network deployments use specialized and highly optimized protocols that require the presence of a gateway that connects the sensor network to the outside world. The gateway must be tailored to the specific protocols used inside the sensor network. To be able to avoid the use of a specialized gateway, several recent systems use the IP protocol inside the sensor network [3, 7, 11]. Running IP inside the sensor network has the benefit of interoperability at the network layer. IP does not, however, automatically enable integration at the higher layer.

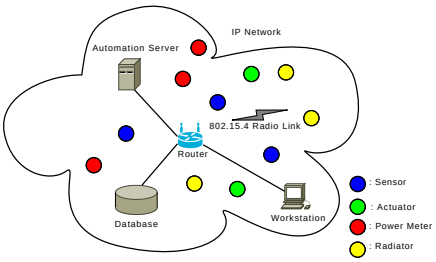

Figure 1. Sensors, actuators, and other energy management devices such as energy meters and radiators are part of the same IP network as the automation server, and the database servers that carry the data on which the automation system operates.

To integrate sensor networks with existing IT systems, the use of Web services has been proposed [9, 6, 11]. Web services are a mechanism that is widely used in general purpose IT systems, such as business logic systems and data bases. Web services provide a structured and interoperable mechanism for data acquisition, data storage, and data replication both within and outside of the sensor network. A Web services-based sensor network can be integrated into office automation or home energy management systems that are built on standard IT system components. Unlike specialized gateway-based approaches, Web services provide an architecture that is able to evolve as the field grows.

Our overall architecture is shown in Figure 1. Sensors, actuators, and other devices, are part of the same IP network as automation manager software as well as the database servers that hold energy consumption data history. The servers can communicate directly with the devices using the Web services mechanism.

The contribution of this paper is twofold. First, we show the feasibility of using RESTful Web services on an IP-based multi-hop low-power sensor networks. Second, we perform an extensive evaluation where we quantify the performance and power consumption of REST, showing that a REST transaction over a multi-hop low-power network typically is completed within fractions of a second, and with a low power consumption. Furthermore, we quantify the over-

BuildSys'09, November 3, 2009, Berkeley, CA, USA.

head of IPv6 versus an IPv4-based sensor network. To the best of our knowledge, this is the first time that Web services for sensor networks have been evaluated in a multi-hop setting.

2 Related Work

For general-purpose computing, Web services are a well-established mechanism. Web services for general purpose computing has traditionally been SOAP-based, but RESTful systems are emerging. Web services have previously been suggested for use in connecting sensor networks with external networks [6, 9, 11]. Existing work has, however, not investigated the use of Web services extending into the sensor network. Instead, previous efforts have required gateway servers on the border of the sensor network. In contrast, we extend the Web services into the sensor network itself.

TinyREST [9], is developed as part of a Home Services Framework. Its goal is to generate a specific REST based approach for the framework rather than providing a generic framework that this work aims for. Other than IP support, the work also does not include multihop routing and reliability within WSN, both of which are supported in our work, especially reliability is inherently supported thanks to our approach of using standard TCP/IP. A gateway connected to a base station is used to map the set of requests to TinyOS messages and vice versa, which also performs some other tasks such as validity checks.

Priyantha et al [11] have recently showed the feasibility of SOAP-based Web services. Their work revealed several important insights into the interactions between Web services, the underlying TCP protocol, and power-saving MAC and link layer protocols. Our work has three major differences. First, we show that RESTful Web services, a much simpler mechanism than SOAP-based Web services, provide benefits in terms of completion time and power consumption. Second, we integrate the Web services mechanism with an off-the-shelf power-saving MAC protocol (X-MAC) and provide important insights into optimizing its use for RESTful Web services. Third, we provide experimental results from a multi-hop network. To quantitatively compare our system with a SOAP-based Web services mechanism, we have implemented a SOAP-based mechanism and compare the performance in Section 6.

IP-based sensor networks have seen much work in the recent past [4, 7, 11]. Recent work includes using an IP-based sensor network to monitor power consumption in buildings [8]. Our work differs in that we are using an interoperable application layer, RESTful Web services, that can be directly integrated into other IT systems.

3 Web Services for Sensor Networks

Web services are a common name for a set of techniques for developing interoperable distributed applications usually using Web-related standards such as HTTP. Web services are generally categorized in two classes: SOAP-based Web services and RESTful (or REST-based) Web services. SOAP-based Web services employ Simple Object Access Protocol (SOAP) standard. RESTful Web services use Representational State Transfer (REST), a much more lightweight

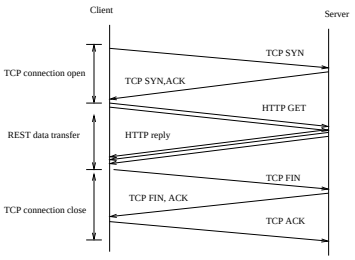

Figure 2. A REST transaction consists of three phases: TCP connection open, REST transaction, and TCP connection closing.

mechanism than SOAP, that provides functionality similar to SOAP-based Web services.

REST is a software architectural style for distributed systems, originally defined by Roy Fielding [5], one of the original designers of the HTTP protocol. REST allows a software system to be distributed over a set of clients and servers, communicating with each other over HTTP. There are several systems using REST, such as the Atom Web syndication protocol used as a news feed protocol by thousands of Web sites, as well as the Sun CloudAPI.

The main abstraction of REST is the resources. Every resource has a URI and using these URIs it is possible to link resources. It is possible to have different representations for the same resource which is a powerful concept, e.g. a server can serve HTML content for human consumption and XML or JSON for machines. REST typically use the standard HTTP request methods.

The network transactions used by a RESTful Web service implementation are simple, as shown in Figure 2. The transaction consist of three phases: the TCP connection open phase, the REST transaction phase, and the TCP connection closing phase. The TCP connection open phase establishes a TCP connection between the client and the server. In our scenario, the client is running on a computer outside the sensor network and the server is running on a sensor network mote. The REST transaction phase, which is initiated only if the TCP connection succeeds, is when the actual data is transmitted. Data can be transmitted both from the client to the server or to the client from the server, depending on why the REST transaction was initiated. Packet loss during the REST transaction phase is handled by the normal retransmission mechanism of TCP. When the REST data has been successfully transmitted, the TCP connection closing phase is performed.

3.1 Data Formats

The data exchanged in a REST transaction can be formatted differently depending on the application. Although XML-based formats frequently are used, they are only one of the many available options. The drawback of the XML-based formats are their size. The XML format is verbose and therefore is not suitable for low power and low data rate sensor networks.

The data format we use in our system is JavaScript Object Notation, JSON. JSON, defined in RFC4627 [2], is a lightweight and language independent text format for inter-

```
{ "Sensors": { "item":
    [ {"name": "Temperature", "value": 26.1},
      {"name": "Light", "value": 87} ] }
}
```

Figure 3. A JSON document.

changing data. JSON serializes data structures, such as numbers and arrays, as strings formatted according to the JSON specification. JSON is more compact than XML as it provides an implicit data structure format. JSON does not require any XML parsing on the sensor network nodes. An example JSON document is provided in Figure 3

3.2 Power-saving MAC Protocols

To save power, sensor nodes must switch off their radios as often as possible [10]. To coordinate the sleep cycles between nodes, nodes use a power-saving MAC protocol. X-MAC [1] is a low power MAC protocol that uses a sequence of short preambles to wake up the receivers. Radio transceiver is the most energy consuming component of a typical sensor node and idle-listening constitutes the main part of total energy usage. X-MAC addresses this problem; motes save energy by switching off the radio most of the time and hence reducing idle radio listening. Nodes wake up for a short time in regular periods to listen for preambles. When a node wakes up and receives a preamble addressed to itself, it replies with an acknowledgement showing that it is awake. Upon reception of the acknowledgement from the receiver, sender transmits the whole packet.

4 A RESTful Sensor Network Architecture

We have designed and implemented a RESTful Web service architecture for sensor motes. Our architecture makes use of two main mechanisms to make Web services a viable alternative for wireless sensor networks: a session-aware power-saving MAC protocol and use of the Conditional HTTP GET mechanism. We now look into each of these mechanisms in detail.

4.1 A Session-Aware Power-Saving MAC Protocol

When interacting with wireless sensor networks, getting good completion times is not enough, energy efficiency is also important. For that purpose, we use the X-MAC duty-cycling MAC protocol [1] as MAC layer protocol. Although X-MAC is being used efficiently in typical WSN applications for some time, to the best of our knowledge the performance of X-MAC as a lower layer for TCP traffic has not been previously studied.

In a typical TCP communication, there appears a continuous traffic in both directions until the connection is closed. This is because TCP is a reliable communication protocol and ACKs are sent to guarantee it even one of the sides does not have any data to transmit. This means that both packets and their corresponding ACKs suffer from the wake-up time imposed by X-MAC.

To improve the performance, we present a session-aware X-MAC. Our session-aware X-MAC lets the radio be switched on during a TCP connection; precisely between the periods of SYN packet reception and FIN packet transmission. This solution decreases the delays significantly since the only packet that suffers from wake-up delay is the first SYN packet.

4.2 Conditional HTTP GET

Conditional HTTP GET is designed to save time and bandwidth by employing certain response (Last-Modified and ETag) and request headers (If-Modified-Since and If-None-Match). The idea is that if the data is not changed after the last time client fetched it, the server can notify client by 304 (Not Modified) status and refrain from sending the data again, thereby saving bandwidth and time. Every time a server sends data, it includes Last-Modified (last time the data was changed) and/or ETag headers (opaque string symbolizing a specific version of data). When the client asks for the same resource later, it provides this information in If-Modified-Since and If-None-Match headers, thereby allowing the server to make a decision whether the resource has changed or not. If it is changed, a response code of 200 (OK) and the new data in the entity-body is served, or else 304 (Not Modified) is returned only, then the client uses its cached data knowing the fact that the underlying data hasn't changed since the first request.

5 Implementation

We have implemented our architecture in the Contiki operating system and with the uIPv4 and uIPv6 IP stacks. We have implemented both our RESTful system and a prototype SOAP-based Web service implementation. The SOAP-based implementation is used as a reference point in the performance evaluation and is not intended for general use. It implements only the necessary mechanisms for receiving data and producing a response.

The implementation is lightweight in terms of memory footprint: the implementation requires only about 4 kilobytes bytes of ROM and a few hundred bytes of RAM. The details of the implementation of modules regarding memory usage is given in Table 1.

6 Evaluation

We evaluate our RESTful Web service architecture for sensor networks using two primary metrics: completion time and power consumption. The completion time is the time between an application on the PC issues a REST call until the reply has been received. The completion time metric includes both the TCP connection open phase, the REST data transaction phase, and the TCP connection close phase, as illustrated in Figure 2.

We quantify the effects of six different mechanisms and scenarios: the effect of the X-MAC power-saving MAC protocol, the effect of introducing session awareness to the power-saving MAC protocol, the effect of the Conditional HTTP GET extension, the effect of multiple network hops, the effect of using REST instead of SOAP-based Web services, and the effect of using IPv6 or IPv4.

Our results show that the use of a power-saving MAC protocol, which reduces the power consumption, increases completion time and that session awareness significantly improves completion times. Furthermore, using Conditional HTTP GET potentially halves the completion time. Multiple hops increase the completion time proportionally to the

Module	Code Size	RAM Footprint
HTTP Server	3976	72
REST Engine	692	4

Module	Code Size	RAM Footprint
HTTP Server	3976	72
XML Parser	5260	4
SOAP Engine	2354	36

Table 1. Memory footprint of our Web services implementations. The RESTful implementation is on the left and the SOAP-based implementation on the right. The SOAP-based implementation includes an XML parser, which is required because XML is mandatory in SOAP.

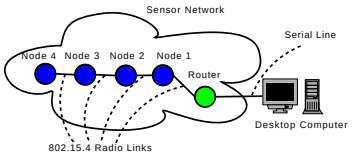

Figure 4. The experimental setup consists of a multi-hop network connected with a serial USB link to a PC workstation.

Web Service	Request Size	Response Size	Total
Dummy	84	48	132
LED Control	89	52	141
Light	79	135	214
Temperature	85	141	226
Sensors	81	324	405

Table 2. Details of the five RESTful Web services. Sizes are given in bytes and do not include TCP/IP and lower layer headers.

number of hops. To provide a baseline to which our performance results can be compared, we provide measurements for our SOAP-based Web service implementation. We primarily use IPv4 in our experiments, but our results show that IPv6 reduces performance but the performance is still similar to that of IPv4.

6.1 Experimental Setup

The experimental setup, shown in Figure 4, consists of a testbed of Tmote Sky motes and a desktop computer running Ubuntu Linux. One mote is used as a router that connects the sensor IP network and the desktop computer. The motes run Contiki. The router mote sends and receives packets to and from the Linux PC using Serial Line IP (SLIP). We use the curl command line tool [12] to provide the workload.

The testbed setup is intentionally simple to avoid irrelevant network effects. We use pre-configured routing tables on every node in all experiments to avoid any effects of a dynamic routing protocol to influence our measurements.

All experiments are repeated 20 times, five times in four sets each, and the average of the results are reported, along with the standard deviation of the results. We use Contiki's build-in power profiling mechanism to obtain power and energy measurements.

We use five different Web service calls for our experiments: one Web service call that does not contain any application data, called Dummy; one Web service call that controls the mote by turning on or off an on-board LED, called LED Control; and three data-acquisition Web service calls that read the on-board Tmote Sky sensors, called Light, Temperature, and Sensors. The details of the request and response sizes of each call are provided in Table 2.

6.2 Session-Aware Power-Saving MAC Protocol

To measure the effects of the session-aware X-MAC protocol, we measure completion times and power consumption both for the original X-MAC and the session-aware X-MAC. We use two different duty cycle configurations of the X-MAC duty cycling radio protocol. We call the two configurations X-MAC and X-MAC2. X-MAC has an off time of

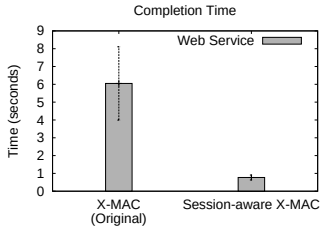

Figure 5. The session-aware X-MAC significantly reduces completion time.

1/4 seconds, resulting in a duty cycle of 2%. X-MAC2 has a 1/2 second off time, resulting in a duty cycle of 1%. The Sensors Web service, shown in Table 2, and X-MAC2 are used to obtain completion times comparison shown in Figure 5.

Session-aware X-MAC outperforms original X-MAC in terms of completion times. This is due to the radio being switched on in between TCP message exchanges. With normal X-MAC, the radio needs to be woken up by a series of strobe packets for every TCP packet and corresponding ACK.

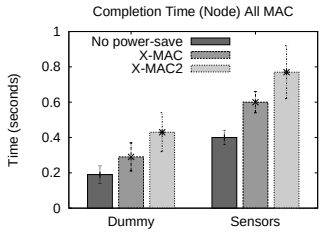

Figure 6. A power-saving MAC protocol significantly reduces power consumption at the price of a higher completion time.. The X-MAC2 configuration (1% radio duty cycle) result in longer completion times than the X-MAC1 configuration (2% radio duty cycle).

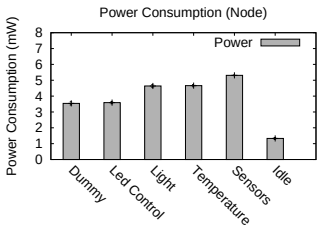

Figure 7. The power consumption of the five Web services calls. Idle power consumption is provided as a reference.

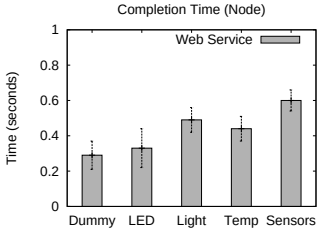

Figure 8. Completion times of Web services on sensor node.

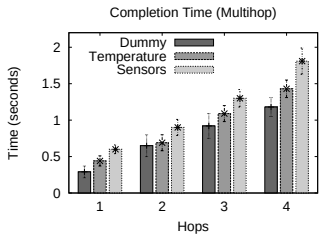

Figure 9. Completion times of Web services over multiple hops.

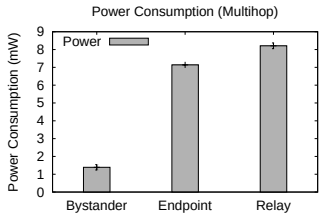

Figure 10. Power consumption of a bystander node, an endpoint node, and a relay node in a multihop network.

6.3 Completion Time and Power Consumption

We evaluate serving RESTful Web services on Tmote Sky motes in terms of power consumption and completion time. Our evaluation confirms that it is reasonable to realize RESTful Web services on wireless sensor networks and that the overheads are reasonable, even with the overhead resulting from TCP/IP and the somewhat verbose nature of HTTP.

Completion times are measured on the desktop PC, using the built-in `time` command of bash shell. The command measures the interval between issuing the Web service call via curl tool [12] and getting the response. Figure 8 shows the results of the measurements of the single-hop who communicates with the router using radio communication. The results show that requests can be fulfilled within a second using a power conserving MAC protocol.

6.4 The Effect of the Conditional HTTP GET

We evaluate the effect in terms of response time and power consumption of the use of the Conditional HTTP GET mechanism. Conditional GET is a caching technique in which the client gets the content from its cache if data is not changed. Two Web service calls, Sensors and Temperature from Table 2, are analyzed using Conditional GET and compared with the original results. The results are given in Table 3.

Cached version have a little bit bigger request data size because of the extra ETag header they transmit, whereas they have significantly smaller response data sizes since they do not include any data content. As it is seen, the response sizes for the cached version are same for both services since the same data is transfered, namely only the headers which happens to be the same for these examples. This also explains why the completion times are roughly same. Also, as expected, cached Sensors Web service performs better in power saving as well as completion time decrease than Temperature service which is consistent to the bandwidth saves.

Web Service	Data Size Reduction	Power Saving	Completion Time Reduction
Temperature	35.4%	24.0%	31.8%
Sensors	64.9%	33.1%	53.3%

Table 3. Performance improvement provided by Conditional GET.

6.5 Results in a Multi-hop Network

In order to evaluate the effect of multi-hop communication for sensor network Web services, we measure completion times of a set of Web services over a multi-hop network. We use the session-aware X-MAC on every hop of the network. Figure 9 shows the measured completion times, with a varied number of hops. The results show that delay caused by relaying RESTful requests in a wireless sensor network is quite reasonable even in a multi-hop network.

Figure 10 shows the power consumption of three nodes in the multi-hop network. The figure shows the power consumption of a bystander node (not serving any Web service nor relaying it), an endpoint node (actually serving the Web service), and a relayer node (Web service is served by the next hop node). The Sensors Web service is used for all measurements. The results show that the power consumption increases for nodes that are either endpoints or relay nodes. Relay nodes have a slightly higher power consumption because the session-aware MAC protocol enables duty cycling some time after the session has been closed by the endpoint node.

6.6 RESTful versus SOAP-Based Web Services

We quantify the overhead of SOAP-based Web services over that of REST by comparing the RESTFul LED control

Web Service	Request Size	Response Size	Total
LED Control	576	498	1074

Table 4. The request and response sizes, measured in bytes, for the SOAP-based LED control Web service.

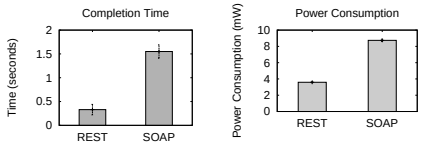

Figure 11. A RESTful Web service outperforms the equivalent SOAP-based Web service in power consumption as well as completion time.

Web service and the SOAP-based equivalent.

The request and response size of the SOAP-based LED control Web service is given in Table 4. The resulting power consumption and completion time are given in Figure 11. The SOAP-based Web service consumes almost twice the power as the RESTful approach. The SOAP-based Web service has nearly four times longer completion times.

6.7 IPv4 versus IPv6

In recent work on IP-based sensor networks and Web services for sensor networks, both IPv4 and IPv6 has been used [4, 7, 8, 11]. No one has previously provided a quantitative comparison between IPv4 and IPv6, however.

To quantify the effect of IPv4 versus IPv6, we run the completion time experiment with the Dummy Web services call and the Sensors call, using both IPv4 and IPv6. We use the uIPv6 implementation in Contiki and 6lowpan header compression [4]. To avoid inadvertently measuring effects caused by a power-saving MAC protocol, for this experiment we do not use a power-saving MAC protocol.

The results of the IPv6-based experiment are shown in Figure 12. Although IPv6 increases the completion times over that of IPv4, the completion times is still below one second.

6.8 Battery Lifetime

Armed with the data from the above experiments, we can use the information to compute an estimate of the battery lifetime. We estimate battery lifetime of a sensor node serving a typical sensor monitoring service: temperature service in Table 2. We assume two AA batteries offering 2500 mAh each. With this data, the estimated battery life of the sensor node over the number of calls is as in Figure 13.

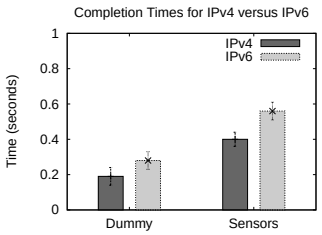

Figure 12. The completion time increases with IPv6 versus IPv4 due to header overhead.

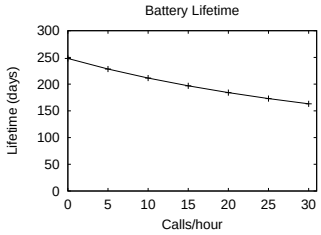

Figure 13. Estimated battery lifetime of two 2500 mAh batteries over the number of Web service calls

7 Conclusions

Sensor networks are seen as an important technology for emerging energy management systems for buildings, offices, and the smart grid, but their integration into existing IT systems are still an open question. We present a RESTful Web service architecture for sensor networks that allow direct integration between Web service-based IT systems and IP-based sensor networks. Our results show sub-second completion time of Web service requests to low-power nodes in both single-hop and multi-hop networks.

8 Acknowledgments

This work was supported by SSF, VINNOVA, CONET, and GINSENG.

9 References

[1] M. Buettner, V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *SenSys 2006*.

[2] D. Crockford. The application/json media type for javascript object notation (json). Internet RFC 4627, July 2006.

[3] A. Dunkels and J-P Vasseur. IP for Smart Objects, September 2008. IPSO Alliance White Paper 1, available from www.ipso-alliance.org.

[4] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *SenSys 2008*.

[5] R. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000.

[6] D. Guinard and V Trifa. Towards the web of things: Web mashups for embedded devices. In *Proceedings of WWW (International World Wide Web Conferences)*, Madrid, Spain, 2009.

[7] J. Hui and D. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proceedings of the 6th international Conference on Embedded Networked Sensor Systems*, Raleigh, North Carolina, USA, November 2008.

[8] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler. Design and implementation of a high-fidelity ac metering network. In *Proceedings of IPSN 2009*, April 2009.

[9] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim. TinyREST - a protocol for integrating sensor networks into the internet. *in Proc. of REALWSN*, 2005.

[10] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. IPSN/SPOTS'05*, Los Angeles, CA, USA, April 2005.

[11] B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao. Tiny web services: design and implementation of interoperable and evolvable sensor networks. In *Proceedings of SenSys 2008*, 2008.

[12] D. Stenberg. Curl web site. http://curl.haxx.se/. Accessed on May 8, 2009.